Autonomous learning of parameters in differential equations
Adel Mezine, Artémis Llamosi, Véronique Letort, Michèle Sebag, Florence d’Alché-Buc

To cite this version:
Adel Mezine, Artémis Llamosi, Véronique Letort, Michèle Sebag, Florence d’Alché-Buc. Autonomous learning of parameters in differential equations. 32nd International Conference on Machine Learning (ICML) - AutoML workshop, Jul 2015, Lille, France. hal-01272370

HAL Id: hal-01272370
https://hal.science/hal-01272370
Submitted on 10 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Autonomous learning of parameters in differential equations

Adel Mezine
Université d’Evry Val d’Essonne, IBISC, Evry, France
adel.mezine@ibisc.fr

Artémis Llamosi
Université de Paris Diderot, MSC, CNRS UMR 7057, Paris, France
artemis.llamosi@univ-paris-diderot.fr

Veronique Letort
Ecole Centrale Paris, MAS, Châtenay-Malabry, France
veronique.letort@ecp.fr

Michèle Sebag
Université de Paris-Sud, LRI, CNRS UMR 8623, Orsay, France
michele.sebag@lri.fr

Florence d’Alché-Buc
Ecole Telecom ParisTech, LTCI, CNRS UMR 5141, Paris, France
florence.dalche@telecom-paristech.fr

Abstract
We propose EDEN+ a fully automatic learner of parameters in dynamical systems that selects automatically the next experiment to do in the laboratory to improve its performance. EDEN+ improves upon EDEN, an experimental design algorithm proposed in the context of DREAM 6 and 7 challenges, with several new features: ability to take into account experiments with different costs, Monte-Carlo Tree Search parallelization, global optimization for parameter estimation. An illustration of the behaviour of EDEN+ is given on one of the DREAM7 challenging problems.

Keywords: active learning, system identification, Monte-Carlo Tree Search, multi-armed bandits, gene regulatory networks, differential equations

1. Motivation
Discovery in science relies on the choice of appropriate experiments whose results allow to refute or confirm working hypotheses. Machine learning can provide a precious help in the laboratory not only by automatically analyzing the data at hand but also by making suggestions to the experimenter about which data to acquire as already shown in King et al. (2004). This is especially relevant in a biology lab where the cost of experiments is very often prohibitive. Taking the example of dynamical modeling in systems biology, we address the general problem of parameter estimation of differential equations together with the experimental design in a sequential manner. Starting from an initial system of parametric ordinary differential equations and an initial dataset, the EDEN+ algorithm tackles this active learning problem as one player game where each move is a choice of a single experiment (usually a perturbation experiment). The game is won if the parameters and the hidden states are estimated with sufficient accuracy.

1. Experimental Design for parameter Estimation in a Network

2. Problem description

We consider a dynamical system whose state at time $t$ is the $d$-dimensional vector $x(t) = [x_1(t) \ldots x_d(t)]$ and whose dynamics are modeled by first-order ODE governed by a function $f$ depending on parameters $\theta$ and on an exogenous input $u(t)$.

We partially observe its behavior given some initial condition $x(0) = x_0$ and some neutral input $u(t) = g_0(t)$, e.g. without any intervention (as defined below). Let $H$ be the observation model, typically a projection of $\mathbb{R}^d$ in a lower dimensional space $\mathbb{R}^p$ ($p < d$), $Y_0 = (y_{tk})_{k=0,\ldots,n-1}$, a time series of $n$ $p$-dimensional observations and $(\epsilon_{tk})_{k=0,\ldots,n-1}$, $n$ i.i.d realizations of a $p$-dimensional noise. For sake of simplicity, $y_{tk}$ (resp. $\epsilon_{tk}$) will be noted $y_k$ (resp. $\epsilon_k$). Given these assumptions, the observations and the states of the system in Quach et al. (2007) are expressed as follows: given $k = 0, \ldots, n-1$:

$$
\begin{align*}
  x(0) &= x_0 \\
  x(t_{k+1}) &= x(t_k) + \int_{t_k}^{t_{k+1}} f(x(\tau), u(\tau), \theta) d\tau \\
  y_k &= H(x(t_k), u(t), \theta) + \epsilon_k \quad (1)
\end{align*}
$$

This model is a state-space model with the particularity that the hidden process is deterministic and computed using a numerical integration. Although nonlinear filtering approaches such as Unscented Kalman Filtering (UKF) in Quach et al. (2007) and extended Kalman Filtering (EKF) in Wang et al. (2009) can be applied, we will use here global maximization algorithm of the log-likelihood. A major difficulty in parameter and hidden state estimation is the practical non-identifiability of parameters. Namely, two different parameter solutions can provide the same likelihood value. A well-known way to address this issue is to intervene on the dynamical system to perform additional experiments producing observations that exhibit different kinetics. It can consist either in perturbing the system, e.g. forcing the level of a state variable to be zero, or in changing the observation model by allowing to observe different state variables. To benefit from these new data during the estimation phase, the ODE model must account for all the available experiments defined by a finite set of size $E$: $\mathcal{E} = \mathcal{E}_0 = \{e_1, \ldots, e_E\}$. This can be done by defining adequately the exogenous input $u(t)$ among a set of intervention functions $g_{e}(t), e \in \mathcal{E}$ as shown in the application section.

Automatic selection of the appropriate interventions (experiments) to apply to the system is the purpose of this work. We are especially interested in an active learning algorithm that sequentially selects at each step $\ell$, the next experiment $e^*_\ell$ among the candidate experiments of the set $\mathcal{E}_\ell = \mathcal{E}_{\ell-1} - \{e^*_\ell-1\}$, that will produce the most useful dataset for the estimation task. Contrary to the purely statistical approaches of experimental design, ours aims at offering the possibility to anticipate on the fact that one given experiment will be followed by others. The search for an optimal $e^*_\ell \in \mathcal{E}_\ell$ thus depends on the potential subsequent sequences of experiments, their total number being limited by a finite affordable budget to account for the cost of real experiments. In this work, we improve a previous version of the active learning algorithm EDEN presented in Llamosi et al. (2014) by increasing considerably its autonomy and its scope of application.
3. Algorithm

As described in Llamosi et al. (2014) the EDEN algorithm consists mainly in three parts: the estimation, the design of experiment (DOE) and the experimentation. The estimation step aims at exploiting the current available data to learn the parameters of the model. The design of experiment is automatically performed by an algorithm of the Monte-Carlo Tree Search (MCTS) family that suggests a sequence of one or more experiments. The experimentations are done with respect to the recommendations of the DOE procedure. Algorithm 1 presents the improved version of EDEN.

Data: Budget: $B$; Initial data: $D_0$; Design space: $E$; Initial candidate parameters: $\Theta_0$.

Algo: Learner; Reward.

Input: Intermediate budget: $b$; Initial model: $M_0$; Cost function: $\text{error}()$; Tree policy: $\phi$; Recommendation: $\psi$; Number of tree walks: $N$; Maximum tree depth: $T$; Size of the version space: $N_{VS}$; Rejection threshold: $\lambda > 1$.

Output: $\hat{\Theta}$.

$\text{budget} \leftarrow B$
$D \leftarrow D_0$
$M \leftarrow M_0$

while $\inf_{e \in E} \text{price}(e) \leq \text{budget}$ do

$VS = \emptyset$ /* Building an empirical version space */

while $\text{card}(VS) < N_{VS}$ do

$\hat{\Theta} \leftarrow \text{Learner}(\Theta_0, M, D, \text{error}())$

$VS \leftarrow VS \cup \left\{ \theta \in \hat{\Theta} \mid \forall (M_i, D_i) \in M \times D, \text{error}(M_i, D_i, \theta) < \lambda \cdot \min_{\theta'}(\text{error}(M_i, D_i, \theta')) \right\}$

$\Theta_0 \leftarrow \hat{\Theta}$

end /* Design of experiment based on Monte-Carlo Tree Search */

$e \leftarrow \text{MCTS}(\phi, \psi, N, E, VS, \text{Learner}, \min(b, B), \text{horizon}, \text{reward}())$

$D_e \leftarrow \text{Experimentation}(e)$; // In silico or in vivo experiment

$M_e \leftarrow \text{Modeling}(e)$

$D \leftarrow D \cup D_e$

$M \leftarrow \text{Fusion}(M, M_e)$

$E \leftarrow E \setminus \{e\}$; // Replicated experiment are not considered

$\text{budget} \leftarrow \text{budget} - \text{price}(e)$

end

$\hat{\Theta} \leftarrow \text{Learner}(\Theta_0, M, D, \text{error}())$

return $\hat{\Theta}$

Algorithm 1: EDEN+

The EDEN+ algorithm requires a finite set of experiments $E$ that are feasible in real world and can be simulated by an appropriate model. A cost is associated to each experiment, and a finite budget $B > 0$ is available. A prior is required to initialize the EDEN+ algorithm. Depending on the choice of the estimation algorithm, the prior can be formulated either as a hypercube, that delimits the region of parameters to explore, a probability distribution or a population of parameters. At each step the Learner updates this prior $\Theta_0$.

The estimation phase aims at exploring the parameter space in order to extract a finite subset of plausible parameter vectors called Version Space. It is essential that this set highlights any non-identifiability issues that may exist. Furthermore, it should exhibit a wide spectrum of behaviours with respect to the next experiments that can be considered. In
order to solve the parameter estimation problem, we propose to use the global optimization algorithm Cooperative Enhanced Scatter Search (CESS) introduced in Villaverde et al. (2012). CESS is a metaheuristic approach that has shown good performances in various problems as mentioned in Villaverde et al. (2015). The term Cooperative indicates that we refer to the parallel implementation in which several threads run ESS in parallel and synchronize regularly.

During the step of experimental design, the MCTS algorithm is applied to explore efficiently the set of sequences of experiments from $\mathcal{E}$ thanks to a game tree. An intermediate budget $b$ and a horizon $h$ are required to limit the depth of the tree search. A policy $\phi$ is used to determine which experiment to select. In Kocsis and Szepesvári (2006), the Upper Confident Bound for Tree (UCT) uses the Upper Confident Bound (UCB) algorithm as selection policy to deal with the exploration-exploitation trade-off. The version of MCTS presented in Algorithm 2 has been adapted to allow considering that different experiments can have different costs and thus stop the expansion of the tree search once the budget is exhausted.

The parallelization is an efficient way to increase the power of the MCTS algorithm. Several parallelization methods have been investigated in Gelly et al. (2008) and Chaslot et al. (2008). There are mainly two approaches to parallelize MCTS. The first method is to use a shared memory to store one version of the tree, and several threads explore the tree to update it. The second method that has been chosen here (due to the way Matlab implements multi-threaded applications) consists in building a tree per thread and synchronizing them via messages. In our case, the time to compute the reward is larger than the time to update

\begin{algorithm}
\textbf{Input:} Policy: $\phi$; Recommendation: $\psi$; Number of tree walks: $N$; Design space: $\mathcal{E}$; Version Space: $V.S$; Learner; Intermediate budget $b$; horizon $h$; Reward
\textbf{Output:} Recommended experiment: $e$
\begin{verbatim}
tree = \{root\}
treewalk = 1
while treewalk < N do
  $A \leftarrow \mathcal{E}$
  $path \leftarrow \{root\}$
  $node \leftarrow root$
  while $(price(path) < b) \land (length(path) < horizon)$ do
    $a \leftarrow \phi(tree, path, A)$
    $node \leftarrow getNode(tree, path, a)$; // Void if the action has not associated node
    if $node = \emptyset$ then
      $node \leftarrow tree.addNode(path, a)$; // Add a child associated to the action
    end
    $A \leftarrow A \backslash \{a\}$; // Replicated experiment are not considered
    $path \leftarrow path \cup node$
  end
  $E \leftarrow getAction(tree, path, \mathcal{E})$
  $r \leftarrow Reward(E, V.S, \mathcal{E}, Learner)$
  /* Back-propagation of the reward through the visited path */
  tree.backPropagate(path, r)
end
return $e \leftarrow \psi(tree, \mathcal{E})$
\end{verbatim}
\textbf{Algorithm 2:} MCTS: Monte-Carlo Tree Search

The parallelization is an efficient way to increase the power of the MCTS algorithm. Several parallelization methods have been investigated in Gelly et al. (2008) and Chaslot et al. (2008). There are mainly two approaches to parallelize MCTS. The first method is to use a shared memory to store one version of the tree, and several threads explore the tree to update it. The second method that has been chosen here (due to the way Matlab implements multi-threaded applications) consists in building a tree per thread and synchronizing them via messages. In our case, the time to compute the reward is larger than the time to update
the tree, thus the synchronization can be done frequently. In this way, the decisions taken by the algorithm are always based on the most recent information.

4. Experimental result

EDEN+ has been confronted with the best competitor of the sub-challenge of parameter inference in Dream7 Challenge, namely the orangeballs team in Meyer et al. (2014). Given the topology and the underlying mechanistic model of an *in silico* 9-gene regulatory network, the participants are asked to estimate the parameter vector and to predict the concentrations of some proteins in specific conditions. The full model is given under the form of an ordinary differential equation system that describes the evolution of the protein and mRNA concentration in time associated to each gene.

The performance of the participants is evaluated using two scores $D_{\text{param}}$ and $D_{\text{pred}}$ that are respectively based on the parameter vector estimated and the protein timecourse prediction. $D_{\text{param}}$ represents the error in order of magnitude between the true vector parameter $\theta^*$ and the estimated $\hat{\theta}$.

$$
D_{\text{param}}(\theta, \theta^*) = \frac{1}{N_\theta} \sum_{i=1}^{N_\theta} \left[ \log \left( \frac{\hat{\theta}_i}{\theta^*_i} \right) \right]^2.
$$

(2)

$D_{\text{pred}}$ is the normalized mean squared error between the prediction of three timecourse protein and the trajectories simulated with the true parameters for a given set of perturbations.

$$
D_{\text{pred}}(\theta, \theta^*) = \frac{1}{90} \sum_{k=3,5,8} \sum_{i=1}^{40} \frac{p_k(\hat{\theta}, t_i) - p_k(\theta^*, t_i)}{0.01 + 0.04 \cdot p_k(\theta^*, t_i)^2}.
$$

(3)

In order to apply MCTS, a reward function is designed from the two scores to evaluate the utility of the sequence of experiments explored. The reward consists in multiplying the mean of reduction of each component of $D_{\text{param}}$ and $D_{\text{pred}}$ computed between a surrogate oracle $\theta^* \in V S$ and $\theta^{\text{opt}}(Y) = \arg \min_{\theta \in \mathbb{R}^d} \text{error}(M, Y)$ the optimum outcome of the model with respect to the available data $Y$.

$$
\text{reward}(\theta^*, Y_0, Y_1) = \left( 1 - \frac{D_{\text{param}}(\theta^*, \theta^{\text{opt}}(Y_0 \cup Y_1))}{D_{\text{param}}(\theta^*, \theta^{\text{opt}}(Y_0))} \right) \cdot \left( 1 - \frac{D_{\text{pred}}(\theta^*, \theta^{\text{opt}}(Y_0 \cup Y_1))}{D_{\text{pred}}(\theta^*, \theta^{\text{opt}}(Y_0))} \right).
$$

(4)

Figure 1 shows the evolution of the mean of the scores in the version space. The version space in updated sequentially by adding one dataset at a time to set of observations. The tree is explored with an horizon 3 that makes the algorithm looks ahead to reach a better score in long term.

Table 1 shows the final scores obtained by EDEN+ and DREAM competitors whose methods are described in Meyer et al. (2014). EDEN+ performance is comparable to the first-rank team, with a better prediction score and no human intervention.

5. Implementation

We develop an implementation with the standard version of Matlab. The computation time needed to play the full challenge is one week, using 12 cores of a 4 processors Intel
Figure 1: Evolution of the scores with $h = 3$, $N = 20000$, $N_{VS} = 5000$.

<table>
<thead>
<tr>
<th>Challenger</th>
<th>$D_{param}$</th>
<th>$D_{pred}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>orangeballs</td>
<td>0.0229</td>
<td>0.0024</td>
</tr>
<tr>
<td>Team #341</td>
<td>0.8404</td>
<td>0.0160</td>
</tr>
<tr>
<td>Team #92</td>
<td>0.1592</td>
<td>0.0354</td>
</tr>
<tr>
<td>EDEN+ ($h = 3$)</td>
<td>0.0371</td>
<td>0.0016</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the final scores on Dream 7 Model 1. Only the name of winner orangeballs has been revealed. The results are shown for an horizon of 3 experiments.

Xeon X5670 2.93GHz. The whole EDEN+ is entirely automated for in silico problem. The simulations are performed using a numerical integration algorithm provided by the sundials suite detailed in Hindmarsh et al. (2005). The optimization part of parameter inference problem is performed by the MEIGO toolbox by Egea et al. (2014).

6. Conclusion

A very general algorithm has been developed for dynamical systems identification and experimental design. One of the most important features of this algorithm is its ability to take into account the fact that an experiment is followed by other experiments and thus to anticipate. Allowing for different experiment costs opens the door to real applications and parallelism is helpful to reduce the computational load of exploration and estimation. Working perspectives concern the extension of the approach to other learning problems related to system biology.
References


